# FPGA enclaves

A presentation on how to enable secure remote computation on cloud-based FPGAs.

Niklas Lindskog / Håkan Englund
Ericsson Research
2020-05

# About the authors

Håkan Englund

Senior Security Researcher at
Ericsson Research

PhD in cryptology
Background in platform security, trusted executions environments.

Prior publications: https://www.researchgate.net/scientific-contributions/70123061_Hakan_Englund

hakan.englund@ericsson.com

Niklas Lindskog

Experienced Security Researcher at
Ericsson Research

M Sc. Eng.
Background in military grade embedded systems

First IEEE publication

niklas.lindskog@ericsson.com

# FPGA acceleration in cloud environments

- **Where can you find cloud-based FPGAs?**
  - Big cloud providers, e.g. AWS, Azure and Baidu are offering FPGA acceleration.
  - Offered as everything from acceleration-as-a-service to directly programming the FPGA (FPGA-as-a-service).

- **What is FPGA-as-a-service (FaaS)?**
  - FPGA-as-a-service provides the client with almost the same possibilities as owning the FPGA.
  - Cloud provider controls part of device, "shell", at all times.
    Keeps control of security & management components.
  - Client is assigned a "role", containing programmable logic and some peripherals.
  - Currently, most FaaS devices available are acceleration cards.
    Industry are moving towards system-on-chip devices, i.e. FPGAs with processing subsystems / hard processors.

# Security for FPGA-as-a-service (FaaS)?

There are several security concerns regarding current FPGA-as-a-service solutions:

- After uploading bitstream, the client does not know the state of the device.
  - **Current solution:** Client must trust cloud provider to provide correct information.

- Malicious bitstreams, e.g. causing short circuits, can harm FPGA.
  - **Current solution:** Part of tool chain is owned by cloud provider.
    Client must expose design to cloud provider.
    Cloud owner can inspect bitstream prior to deployment.

- Poor multi-tenancy support in FPGAs.
  Partial reconfiguration capabilities are not designed for a multi-tenancy use case.
  - **Current solution:** No solution. One client per FPGA.

**Bottom line:** Security features on contemporary FPGAs are not designed for cloud usage.
Client needs to trust cloud provider with both data and IP.

# Our solution – in a nutshell

- **Goal –** Create model which removes the need for trust between cloud provider and client.

- The chip manufacturer (CM) act as root-of-trust for both client and cloud provider.
- CM controlled shell owns part of FaaS device and enables the creation of enclave areas in the programmable logic.
- An external inspection service, either owned by CM or trusted third party, inspects client bitstreams for dangerous constructs.
- Client bitstream is only exposed to the CM and the inspection service.
- Client data is only exposed inside an enclave area which is setup by CM and controlled by the client.

- The FaaS device can be booted into one of two modes, **normal** and **enclave** mode.
  - **Enclave mode** enables remote confidential computation.
    CM controls bitstream loader and security-critical peripherals.
    Cloud provider has limited control over device.
  - **Normal mode** enables the current security model.
    Cloud provider controls device.
    CM controls access to device-unique key.

# Our solution – Enclave mode

**Framework changes**

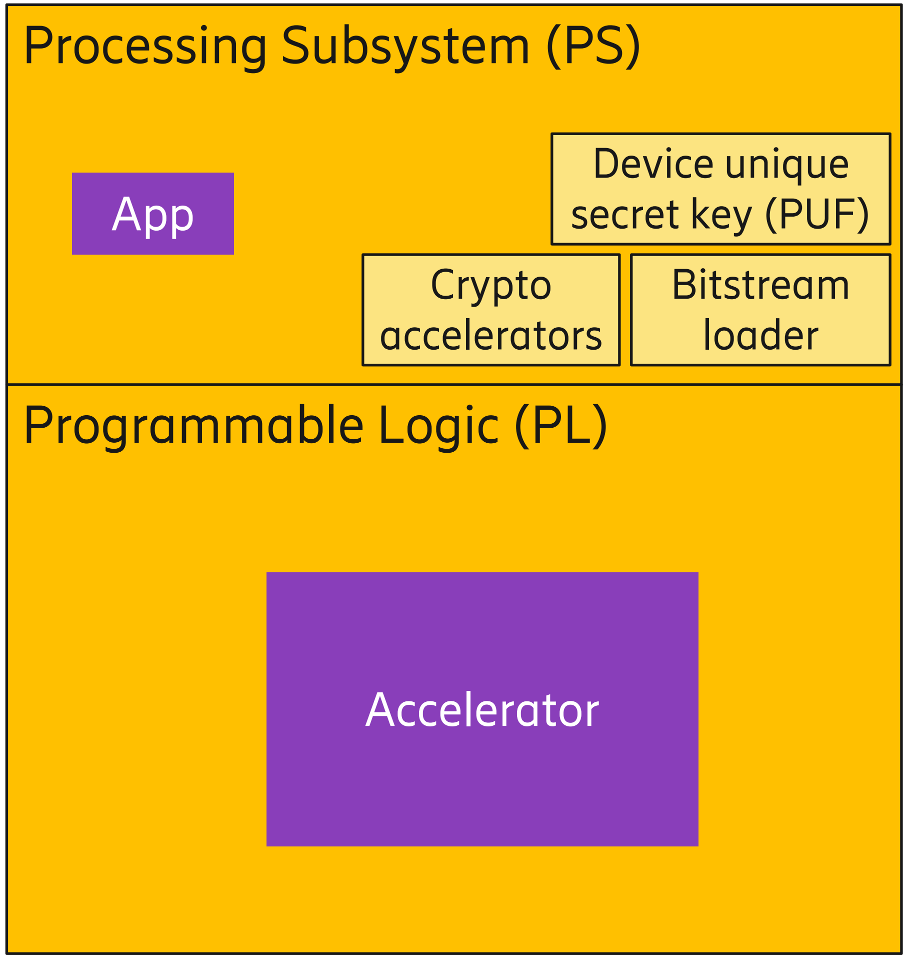
- During manufacturing, CM creates a certificate based on device-unique key and provides it to the FaaS device.
- All booted components must be signed by CM (in normal mode, only first stage bootloader must be signed by CM).
- CM-signed bitstream setup programmable logic to have one or several “enclaves areas”.

**Remote confidential computation**

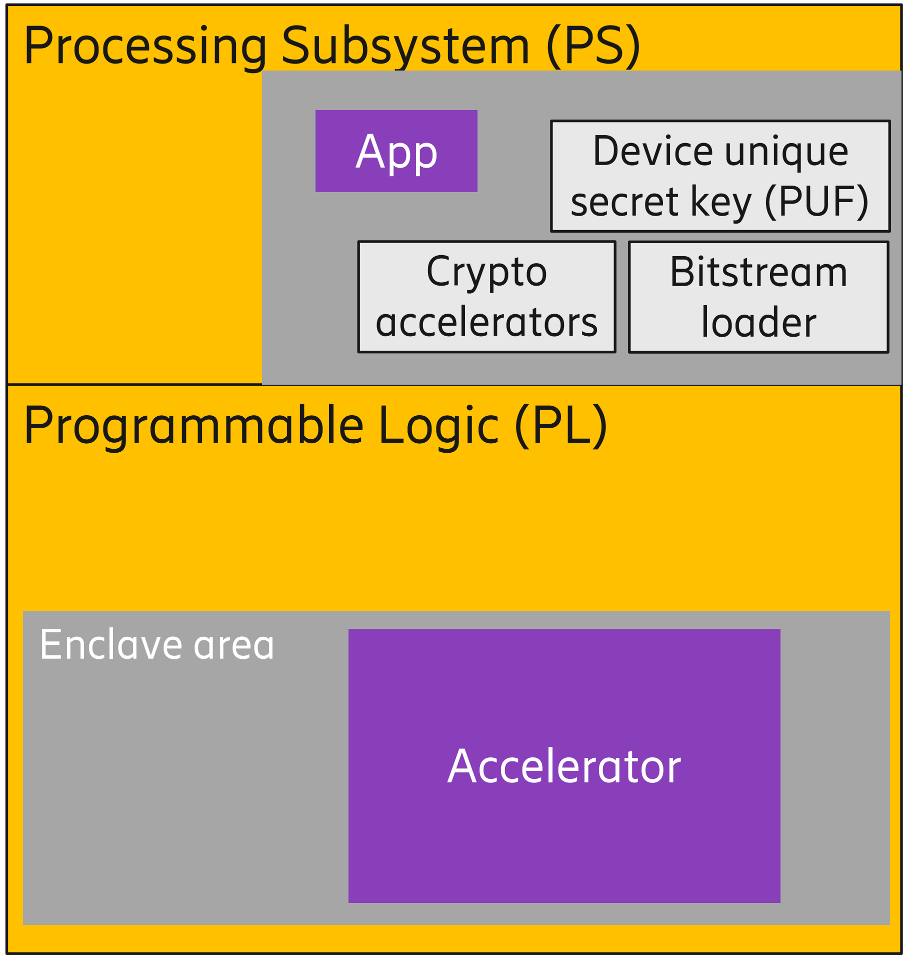
- Client uses own tool chain to create bitstream. External trusted third party/CM inspects, encrypts and signs the bitstream.
- Client provides encrypted bitstream and possibly software applications to the device.
- Cloud provider can inspect the signature and get a guarantee that the bitstream is non-malicious without inspecting the contents.

- User bitstream and software applications are deployed and run in CM-controlled environment.
- Client inspects state of enclave prior to revealing any data.
- Client sets up a session key derived from device unique key.
- When client is finished, the enclave is destroyed.

## FaaS security - today




## FaaS security – our solution




= Controlled by cloud provider

= Controlled by chip manufacturer

= Controlled by client

# Security for FPGA-as-a-service (FaaS)!

- After uploading bitstream, the client does not know the state of the device.
  - ~~**Current solution:** Client must trust cloud provider to provide correct information~~
  - **Our Solution:** Let chip manufacturer (CM) be root-of-trust for both cloud provider and client.
    No mutual trust between cloud provider and client needed.
    CM informs client about device state prior to data exposure.

- Malicious bitstream e.g. (short circuits) can harm FPGA.
  - ~~**Current solution:** Part of tool chain is owned by cloud provider.~~
  - **Our solution:** Chip manufacturer or mutually trusted third party inspects and signs bitstream.

- Poor multi-tenancy support in FPGAs.
  Partial reconfiguration capabilities are not designed for a multi-tenancy use case.
  - ~~**Current solution:** No solution. One client per FPGA.~~
  - **Our solution:** Isolated pre-defined areas (enclaves) which can be populated by different clients.

# Why is it important to move trust from cloud provider to manufacturer?

- **Easier trust model**
  Already implicit trust in the manufacturer when running on hardware.

- **Protects against hackers & cloud insiders**
  A hacker/malicious employee can not use misconfiguration or root access to expose secrets.

- **Enables cloud usage for high security use cases**
  Medical records and other sensitive data is never exposed outside enclave.

- **Enables secure multi-tenancy**
  Isolated, pre-defined programmable logic areas can be used by different clients.
  Bitstream are vetted against dangerous constructs which could be used for eavesdropping.

# And how did we do it?

- Model implemented and tested on Xilinx Zynq Ultrascale MPSoC.

- First stage bootloader inspects signature on all components and informs platform management unit (PMU).

- Hypervisor controls processing subsystem.
  - Applications in host domain handles loading of bitstream, signature checking and network passthrough.
  - Client applications are run in subdomains with separate memory regions.

- Platform management unit (PMU) controls bitstream loader and PUF.
  - PUF is used to create device unique key.

- Programmable logic is setup with bitstream having fixed reconfigurable enclave areas.
  - Enclave deployments handles both key establishment and decryption/encryption.

Thank you for listening!